## DETAILED ACTION

### *Response to Amendment*

1. This Office Action has been issued in response to the amendment filed on 24 September 2009. Claims 29-31 and 55-57 are cancelled. Claims 1-28, 32-54, and 58-73 are pending. Applicants' arguments have been carefully and respectfully considered in light of the amendments and are not persuasive, as they relate to the claim rejections under 35 U.S.C. 102 and 103, as will be discussed below. Accordingly, this action has been made FINAL.

### *Claim Rejections - 35 USC § 102*

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

**3. Claims 1-4, 10, 11, 15, 22, 32, 33-36, 44, 45, 49, 58, and 61-73 are rejected under 35 U.S.C. 102(e) as being anticipated by Burns et al (US 6,925,515 B2).**

With respect to independent claim 1, Burns teaches a method of operating a network file server computer for providing clients with concurrent write access to a file, the method comprising the network file server computer responding to a concurrent write request from a client by:

a) obtaining a lock for the file *(see column 8, lines 49-67)*; and then

b) preallocating a metadata block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

c) releasing the lock for the file *(see column 10, lines 14-30; Note that one the file has been allocated it is unlocked for write.)*; and then

d) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

e) obtaining the lock for the file *(see column 10, lines 31-44)*; and then

f) committing the metadata block to the file in the data storage *(see column 10, line 65 through column 11, line 5)*; and then

g) releasing the lock for the file *(see column 11, lines 6-14)*.

With respect to dependent claim 2, Burns further teaches a hierarchy of blocks including an inode block of metadata, data blocks of file data, and indirect blocks of metadata, and wherein the metadata block for the file is an indirect block of metadata *(see column 10, line 14 through column 11, line 14)*.

With respect to dependent claim 3, Burns further teaches copying data from an original indirect block of the file to the metadata block for the file, the original indirect block of the file having been shared between the file and a read-only version of the file *(see column 10, line 14 through column 11, line 14)*.

With respect to dependent claim 4, Burns further teaches concurrent writing for more than one client to the metadata block for the file *(see column 10, line 14 through column 11, line 14)*.

With respect to dependent claim 10, Burns further teaches writing the metadata block to a log in storage of the network file server computer for committing the metadata block for the file *(see column 10, line 14 through column 11, line 14).*

With respect to dependent claim 11, Burns further teaches gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file *(see column 10, line 14 through column 11, line 14).*

With respect to independent claim 15, Burns teaches a method of operating a network file server computer for providing clients with concurrent write access to a file, the method comprising the network file server computer responding to a concurrent write request from a client by:

a) preallocating a metadata block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

b) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

c) committing the metadata block to the file in the data storage *(see column 10, line 65 through column 11, line 5)*;

wherein the method includes gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata

blocks for the plurality of client write requests to the file by obtaining a lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file *(see column 10, line 14 through column 11, line 14)*.

With respect to dependent claim 22, Burns further teaches processing multiple concurrent read and write requests by pipelining the requests through a first processor and a second processor, the first processor performing metadata management for the multiple concurrent read and write requests, and the second processor performing asynchronous reads and writes for the multiple concurrent read and write requests *(see column 10, line 14 through column 11, line 14)*.

With respect to independent claim 32, Burns teaches A method of operating a network file server computer for providing clients with concurrent write access to a file, the method comprising the network file server computer responding to a concurrent write request from a client by executing a write thread, execution of the write thread including:

a) obtaining an allocation mutex for the file *(see column 10, lines 14-30)*; and then

b) preallocating new metadata blocks that need to be allocated for writing to the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

c) releasing the allocation mutex for the file *(see column 10, lines 14-30)*; and then

d) issuing asynchronous write requests for writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*;

e) waiting for callbacks indicating completion of the asynchronous write requests *(see column 10, lines 14-30)*; and then

f) obtaining the allocation mutex for the file *(see column 10, lines 31-44)*; and then

g) committing the preallocated metadata blocks *(see column 10, line 65 through column 11, line 5)*; and then

h) releasing the allocation mutex for the file *(see column 11, lines 6-14)*.

With respect to claims 33-36, claims 33-36 correspond to claims 1-4 and are rejected for the same reasons as set forth in the rejection of claims 1-4 above.

With respect to claims 44-45, claims 44-45 correspond to claims 10-11 and are rejected for the same reasons as set forth in the rejection of claims 10-11 above.

With respect to claim 49, claim 49 corresponds to claim 15 and is rejected for the same reasons as set forth in the rejection of claim 15 above.

With respect to claim 58, claim 58 corresponds to claim 32 and is rejected for the same reasons as set forth in the rejection of claim 32 above.

With respect to independent claim 61, Burns teaches a method of operating a network file server for providing clients with concurrent write access to a file, the method comprising the network file server responding to a concurrent write request from a client by:

a) preallocating a block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

b) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

c) committing the preallocated block *(see column 10, line 65 through column 11, line 5)*;

wherein the network file server also includes an uncached write interface, a file system cache, and a cached read-write interface, wherein the uncached write interface bypasses the file system cache for sector-aligned write operations, and the network file server is programmed to invalidate cache blocks in the file system cache including sectors being written to by the uncached write interface *(see column 10, line 14 through column 11, line 14)*.

With respect to dependent claim 62, Burns further teaches a final step of returning to said client an acknowledgement of the writing to the file *(see column 10, line 14 through column 11, line 14)*.

With respect to claims 63-67, claims 63-67 correspond to claim 62 and are rejected for the same reasons as set forth in the rejection of claim 62 above.

With respect to dependent claim 68, Burns further teaches a final step of saving the file in disk storage of the network file server *(see column 10, line 14 through column 11, line 14)*.

With respect to claims 69-73, claims 69-73 correspond to claim 68 and are rejected for the same reasons as set forth in the rejection of claim 68 above.

***Claim Rejections - 35 USC § 103***

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

**5. Claims 5-9, 12-14, 16-21, 23-28, 37-43, 46-48, and 50-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Burns in view of Marcotte (US 6,449, 614 B1).**

With respect to dependent claim 5, note the discussion of claim 1 above, Burns discloses all of the elements of claim 1, but fails to explicitly recite the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon failing to find an indication of a conflict with a concurrent write to the new block, preallocating the new block, copying at least a portion of the original block of the file to the new block, and performing the partial write to the new block.

However, Marcotte teaches the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon failing to find an indication of a conflict with a concurrent write to the new block, preallocating the new block, copying at least a portion of the original block of the file to the new block, and performing the partial write to the new block *(see column 13, line 35 through column 14, line 43)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the holding queue of Marcotte for the purpose of achieving better performance eliminate the need to suspend the system.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

Therefore, it would have been obvious to combine Marcotte with Burns to obtain the invention as specified in the instant claims.

With respect to dependent claim 6, note the discussion of claim 1 above, Burns discloses all of the elements of claim 1, but fails to explicitly recite the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon finding an indication of a conflict with a concurrent write to the new block, waiting until resolution of the conflict with the concurrent write to the new block, and then performing the partial write to the new block.

However, Marcotte teaches the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon finding an indication of a conflict with a concurrent write to the new block, waiting until resolution of the conflict with the concurrent write to the new block, and then performing the partial write to the new block *(see column 13, line 35 through column 14, line 43)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the holding queue of Marcotte for the purpose of achieving better performance eliminate the need to suspend the system.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

Therefore, it would have been obvious to combine Marcotte with Burns to obtain the invention as specified in the instant claims.

With respect to dependent claim 7, Marcotte further teaches placing a request for the partial write in a partial write wait queue upon finding an indication of a conflict with a concurrent write to the new block, and performing the partial write upon servicing the partial write wait queue *(see column 13, line 35 through column 14, line 43)*.

With respect to dependent claim 8, note the discussion of claim 1 above, Burns discloses all of the elements of claim 1, but fails to explicitly recite checking an input-output list for a conflicting prior concurrent access to the file, and upon finding a conflicting prior concurrent access to the file, suspending the asynchronous writing to the file until the conflicting prior concurrent access to the file is no longer conflicting.

However, Marcotte teaches checking an input-output list for a conflicting prior concurrent access to the file, and upon finding a conflicting prior concurrent access to the file, suspending the asynchronous writing to the file until the conflicting prior concurrent access to the file is no longer conflicting *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

Therefore, it would have been obvious to combine Marcotte with Burns to obtain the invention as specified in the instant claims.

With respect to dependent claim 9, Marcotte further teaches providing a sector-level granularity of byte range locking for concurrent write access to the file by the suspending of the asynchronous writing to the file until the conflicting prior concurrent access is no longer conflicting *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 12, note the discussion of claim 1 above, Burns discloses all of the elements of claim 1, Burns further teaches checking whether a previous commit is in progress after asynchronously writing to the file and before obtaining the lock for the file for committing the metadata block to the file *(see column 10, line 14 through column 11, line 14)*, but fails to explicitly recite upon finding that a previous commit is in progress, placing a request for committing the metadata block to the file on a staging queue for the file.

However, Marcotte teaches upon finding that a previous commit is in progress, placing a request for committing the metadata block to the file on a staging queue for the file *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

Therefore, it would have been obvious to combine Marcotte with Burns to obtain the invention as specified in the instant claims.

With respect to independent claim 13, Burns teaches a method of operating a network file server computer for providing clients with concurrent write access to a file in data storage, the method comprising the network file server computer responding to a concurrent write request from a client by:

a) preallocating a block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

b) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

c) committing the block to the file in the data storage *(see column 10, line 65 through column 11, line 5)*;

Burns fails to explicitly recite the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon finding an indication of a conflict with a concurrent

write to the new block, waiting until resolution of the conflict with the concurrent write to the new block, and then performing the partial write to the new block.

However, Marcotte teaches the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon finding an indication of a conflict with a concurrent write to the new block, waiting until resolution of the conflict with the concurrent write to the new block, and then performing the partial write to the new block *(see column 13, line 35 through column 14, line 43)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the holding queue of Marcotte for the purpose of achieving better performance eliminate the need to suspend the system.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

Therefore, it would have been obvious to combine Marcotte with Burns to obtain the invention as specified in the instant claims.

With respect to dependent claim 14, Marcotte further teaches placing a request for the partial write in a partial write wait queue upon finding an indication of a conflict with a concurrent write to the new block, and performing the partial write upon servicing the partial write wait queue *(see column 13, line 35 through column 14, line 43)*.

With respect to dependent claim 16, note the discussion of claim 15 above, Burns discloses all of the elements of claim 15, Burns further teaches checking whether a previous commit is in progress after asynchronously writing to the file and before obtaining the lock for the file for committing the metadata block to the file *(see column 10, line 14 through column 11, line 14)*, but fails to explicitly recite upon finding that a previous commit is in progress, placing a request for committing the metadata block to the file on a staging queue for the file.

However, Marcotte teaches upon finding that a previous commit is in progress, placing a request for committing the metadata block to the file on a staging queue for the file *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

With respect to dependent claim 17, note the discussion of claim 15 above, Burns discloses all of the elements of claim 15, but fails to explicitly recite the network file server computer responding to concurrent write requests by writing new data for specified blocks of the file to the disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating the specified blocks of the file in the file system cache.

However, Marcotte teaches the network file server responding to concurrent write requests by writing new data for specified blocks of the file to the disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating the

specified blocks of the file in the file system cache *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

With respect to dependent claim 18, Marcotte further teaches the network file server computer responding to read requests for file blocks not found in the file system cache by reading the file blocks from the file system in disk storage and then checking whether the file blocks have become stale due to concurrent writes to the file blocks, and writing to the file system cache a file block that has not become stale, and not writing to the file system cache a file block that has become stale *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 19, Marcotte further teaches the network file server computer checking a read-in-progress flag for a file block upon finding that the file block is not in the file system cache, and upon finding that the read-in-progress flag indicates that a prior read of the file block is in progress from the file system in the disk storage, waiting for completion of the prior read of the file block from the file system in the disk storage, and then again checking whether the file block is in the file system cache *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 20, Marcotte further teaches the network file server computer setting a read-in-progress flag for a file block upon finding that the file block is not in the file system cache and then beginning to read the file block from the file system in disk storage, clearing the read-in-progress flag upon writing to the file block on disk, and inspecting the read-in-progress flag to determine whether the file block has become stale due a concurrent write to the file block *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 21, Marcotte further teaches the network file server computer maintaining a generation count for each read of a file block from the file system in the disk storage in response to a read request for a file block that is not in the file system cache, and checking whether a file block having been read from the file system in the disk storage has become stale by checking whether the generation count for the file block having been read from the file system is the same as the generation count for the last read request for the same file block *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 23, note the discussion of claim 15 above, Burns discloses all of the elements of claim 15, but fails to explicitly recite serializing the reads by delaying access for each read to a block that is being written to by a prior, in-progress write until completion of the write to the block that is being written to by the prior, in-progress write.

However, Marcotte teaches serializing the reads by delaying access for each read to a block that is being written to by a prior, in-progress write until completion of the write to the block that is being written to by the prior, in-progress write *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

With respect to dependent claim 24, note the discussion of claim 15 above, Burns discloses all of the elements of claim 15, but fails to explicitly recite serializing the writes by delaying access for each write to a block that is being accessed by a prior, in-progress read or write until completion of the read or write to the block that is being accessed by the prior, in-progress read or write.

However, Marcotte teaches serializing the reads by delaying access for each read to a block that is being written to by a prior, in-progress write until completion of the write to the block that is being written to by the prior, in-progress write *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

With respect to independent claim 25, Burns teaches a method of operating a network file server computer for providing clients with concurrent write access to a file in data storage, the

method comprising the network file server computer responding to a concurrent write request from a client by:

a) preallocating a metadata block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

b) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

c) committing the metadata block to the file in the data storage *(see column 10, line 65 through column 11, line 5)*;

Burns fails to explicitly recite the network file server computer responding to concurrent write requests by writing new data for specified blocks of the file to the disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating the specified blocks of the file in the file system cache.

Burns also fails to explicitly recite the network file server computer responding to read requests for file blocks not found in the file system cache by reading the file blocks from the file system in disk storage and then checking whether the file blocks have become stale due to concurrent writes to the file blocks, and writing to the file system cache a file block that has not become stale, and not writing to the file system cache a file block that has become stale.

However, Marcotte teaches the network file server computer responding to concurrent write requests by writing new data for specified blocks of the file to the disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating

the specified blocks of the file in the file system cache *(see column 12, line 7 through column 13, line 32)*.

Marcotte also teaches the network file server computer responding to read requests for file blocks not found in the file system cache by reading the file blocks from the file system in disk storage and then checking whether the file blocks have become stale due to concurrent writes to the file blocks, and writing to the file system cache a file block that has not become stale, and not writing to the file system cache a file block that has become stale *(see column 12, line 7 through column 13, line 32)*.

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the waiting list of Marcotte.

The suggestion/motivation for doing so would have been to manage locks and improve the system *(see column 20, lines 35-49)*.

With respect to dependent claim 26, Marcotte further teaches the network file server computer checking a read-in-progress flag for a file block upon finding that the file block is not in the file system cache, and upon finding that the read-in-progress flag indicates that a prior read of the file block is in progress from the file system in the disk storage, waiting for completion of the prior read of the file block from the file system in the disk storage, and then again checking whether the file block is in the file system cache *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 27, Marcotte further teaches the network file server computer setting a read-in-progress flag for a file block upon finding that the file block is not in the file system cache and then beginning to read the file block from the file system in disk storage, clearing the read-in-progress flag upon writing to the file block on disk, and inspecting the read-in-progress flag to determine whether the file block has become stale due a concurrent write to the file block *(see column 12, line 7 through column 13, line 32)*.

With respect to dependent claim 28, Marcotte further teaches the network file server computer maintaining a generation count for each read of a file block from the file system in the disk storage in response to a read request for a file block that is not in the file system cache, and checking whether a file block having been read from the file system in the disk storage has become stale by checking whether the generation count for the file block having been read from the file system is the same as the generation count for the last read request for the same file block *(see column 12, line 7 through column 13, line 32)*.

With respect to claims 37-43, claims 37-43 correspond to claims 5-9 and are rejected for the same reasons as set forth in the rejection of claims 5-9 above.

With respect to claims 46-48, claims 46-48 correspond to claims 12-14 and are rejected for the same reasons as set forth in the rejection of claims 12-14 above.

With respect to claim 50, claim 50 corresponds to claim 16 and is rejected for the same reasons as set forth in the rejection of claim 16 above.

With respect to claims 51-54, claims 51-54 correspond to claims 25-28 and are rejected for the same reasons as set forth in the rejection of claims 25-28 above.

**6. Claims 59 and 60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Burns in view of Xu et al (US 6,324,581 B1).**

With respect to dependent claim 59, note the discussion of claim 58 above, Burns discloses all of the elements of claim 58 but fails to explicitly recite an uncached write interface, a file system cache and a cached read-write interface, and wherein the uncached write interface bypasses the file system cache for sector-aligned write operations.

However, Xu teaches an uncached write interface, a file system cache and a cached read-write interface, and wherein the uncached write interface bypasses the file system cache for sector-aligned write operations *(see column 8, line 36 through column 9, line 39).*

At the time of the invention, it would have been obvious to one of ordinary skill having the teachings of Burns and Marcotte before him or her, to modify the asynchronous writing of Burns to include the uncached write interface of Xu.

The suggestion/motivation for doing so would have been to reduce the loading of data from data movers by using a cached array *(see column 2, lines 57-61).*

With respect to dependent claim 60, Xu further teaches the network file server is further programmed to invalidate cache blocks in the file system cache including sectors being written to by the uncached write interface *(see column 8, line 36 through column 9, line 39).*

***Response to Arguments***

7. Applicants' arguments with respect to objections and rejections not repeated herein are moot, as the respective objections and rejections have been withdrawn in light of the instant amendments. Those arguments that still deemed relevant are now addressed below.

**A. Applicant Argues:**

Burns does not disclose details of how a block of data is allocated to the file, or committed to the file. For example, there is nothing in Burns disclosing that a metadata block is preallocated for the file, and then the file is asynchronously written to, and then the metadata block is committed to the file in data storage, as recited in applicants' claim 1.

Burns teaches away from releasing the lock for the file in step (c) and then asynchronously writing to the file in step (d) and then again obtaining the lock for the file in step (e), because Burns, as set out above, teaches that the file would be exclusively write-locked by the producer lock when a write operation adding a block to the file is performed on the file. See, for example, Burns col. 11, lines 47-49: "With C and P locks, the web servers are not updated until the P lock holder releases the lock, generally on closing the file."

**Response:**

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded because Burns teaches a method of operating a network file server computer for providing clients with concurrent write access to a file, the method comprising the network file server computer responding to a concurrent write request from a client by:

h) obtaining a lock for the file *(see column 8, lines 49-67)*; and then

i) preallocating a metadata block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

j) releasing the lock for the file *(see column 10, lines 14-30; Note that one the file has been allocated it is unlocked for write.)*; and then

k) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

l) obtaining the lock for the file *(see column 10, lines 31-44)*; and then

m) committing the metadata block to the file in the data storage *(see column 10, line 65 through column 11, line 5)*; and then

releasing the lock for the file *(see column 11, lines 6-14)*.

**B. Applicant Argues:**

With respect to applicants' dependent claims 2 and 3, page 3 of the Official Action cites Burns column 10 line 14 through column 11 line 14. However, Burns does not disclose details of this metadata structure such as an indirect block of metadata.

**Response:**

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded because Burns further teaches a hierarchy of blocks including an inode block of metadata, data blocks of file data, and indirect blocks of metadata, and wherein the metadata block for the file is an indirect block of metadata *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation. It is understood that a file contains metadata and that metadata is being allocated as well.)*.

**C. Applicant Argues:**

With respect to applicants' dependent claim 10, page 3 of the Official Action cites Burns column 10 line 14 through column 11 line 14. However, Burns does not disclose writing a metadata block to a log in storage of the network file server for committing the metadata block.

**Response:**

Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded because Burns further teaches writing the metadata block to a log in storage of the

network file server computer for committing the metadata block for the file *(see column 10, line 14 through column 11, line 14).*

**D. Applicant Argues:**

Burns teaches away from gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file.

**Response:**

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded because Burns further teaches gathering together preallocated metadata blocks for a plurality of client write requests to the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file *(see column 10, line 14 through column 11, line 14).*

**E. Applicant Argues:**

With respect to applicants' independent claim 15, as discussed above with reference to applicants' claim 11, it is respectfully submitted that Burns fails to disclose gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file.

**Response:**

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded because Burns teaches a method of operating a network file server computer for

providing clients with concurrent write access to a file, the method comprising the network file server computer responding to a concurrent write request from a client by:

d) preallocating a metadata block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

e) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

f) committing the metadata block to the file in the data storage *(see column 10, line 65 through column 11, line 5)*;

wherein the method includes gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining a lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file *(see column 10, line 14 through column 11, line 14)*.

**F. Applicant Argues:**

With respect to applicants' independent claim 32, see applicants' remarks above with reference to applicants' claim 1. In short, Burns does not disclose details of how a new metadata block is added to the file. Therefore, applicants respectfully submit that Burns does not disclose that new metadata bocks are added to the file by the applicants' recited sequence of the six steps (b), (c), (d), (e), (f), and (g).

With respect to claims 33-36, applicants respectfully traverse for the reasons given above with reference to claims 1-4.

With respect to claims 44-45, applicants respectfully traverse for the reasons given above with reference to claim 10-11.

With respect to independent claim 49, applicants respectfully traverse for the reasons given above with reference to claim 15.

With respect to independent claim 58, applicants respectfully traverse for the reasons given above with reference to claim 32.

**Response:**

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded dues to the response to claims 1-4, 10-11, 15, and 32 above.

**F. Applicant Argues:**

Burns teaches the network file server invalidating client caches when file system locking is used, rather than the network file server invalidating a file system cache of a cached read-write interface when an uncached write interface bypasses the file system cache by performing a sector-aligned write operation.

**Response:**

With respect to Applicant's argument, the argument is not correct and Examiner is not persuaded because Burns teaches a method of operating a network file server for providing clients with concurrent write access to a file, the method comprising the network file server responding to a concurrent write request from a client by:

d) preallocating a block for the file *(see column 10, lines 14-30; Note that the file is allocated before the write takes place making this a pre-allocation.)*; and then

e) asynchronously writing to the file *(see column 10, lines 30-38; Note that the systematic locks cannot write and allocate at the same time thus making the locks only able to write which by definition is asynchronous.)*; and then

f) committing the preallocated block *(see column 10, line 65 through column 11, line 5)*;

wherein the network file server also includes an uncached write interface, a file system cache, and a cached read-write interface, wherein the uncached write interface bypasses the file system cache for sector-aligned write operations, and the network file server is programmed to invalidate cache blocks in the file system cache including sectors being written to by the uncached write interface *(see column 10, line 14 through column 11, line 14)*.

***Conclusion***

**THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to JARED M. BIBBEE whose telephone number is (571)270-1054. The examiner can normally be reached on IFP.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Apu Mofiz can be reached on 571-272-4080. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jared M Bibbee/
Examiner, Art Unit 2161

/Apu M Mofiz/
Supervisory Patent Examiner, Art Unit 2161